Tree Recursion

Order of Recursive Calls

The Cascade Function

Two Definitions of Cascade

Example: Inverse Cascade

Inverse Cascade

Write a function that prints an inverse cascade:

```python
def inverse_cascade(n):
grow(n)
    print(n)
shrink(n)
    print(n)
def _f_then_g(f, g, n):
    if n:
        f(n)
g(n)
grow = lambda n: _f_then_g(grow, print, n//10)
shrink = lambda n: _f_then_g(print, shrink, n//10)
def inverse_cascade(n):
grow(n)
    print(n)
shrink(n)
```
Tree Recursion

Tree-shaped processes arise whenever executing the body of a recursive function makes more than one recursive call.

\[
\begin{align*}
\text{fib}(n): & \quad 0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots, 9,227,465 \\
\text{def fib}(n): & \quad \text{if } n == 0: \text{return } 0 \\
& \quad \text{elif } n == 1: \text{return } 1 \\
& \quad \text{else: return fib}(n-2) + fib(n-1)
\end{align*}
\]

A Tree-Recursive Process

The computational process of fib evolves into a tree structure.

Repetition in Tree-Recursive Computation

This process is highly repetitive; fib is called on the same argument multiple times.

Example: Counting Partitions

The number of partitions of a positive integer \(n\), using parts up to size \(m\), is the number of ways in which \(n\) can be expressed as the sum of positive integer parts up to \(m\) in increasing order.

\[
\begin{align*}
count\_partitions(6, 4) & \quad 2 + 4 = 6 \\
& \quad 1 + 1 + 4 = 6 \\
& \quad 3 + 3 = 6 \\
& \quad 1 + 2 + 3 = 6 \\
& \quad 1 + 1 + 1 + 3 = 6 \\
& \quad 2 + 2 + 2 = 6 \\
& \quad 1 + 1 + 2 = 6 \\
& \quad 1 + 1 + 1 + 2 = 6 \\
& \quad 1 + 1 + 1 + 1 + 1 = 6
\end{align*}
\]

Counting Partitions

The number of partitions of a positive integer \(n\), using parts up to size \(m\), is the number of ways in which \(n\) can be expressed as the sum of positive integer parts up to \(m\) in increasing order.

\[
\begin{align*}
count\_partitions(6, 4) & \quad \text{Recursive decomposition: finding simpler instances of the problem.} \\
& \quad \text{Explore two possibilities:} \\
& \quad \quad \text{Use at least one 4} \\
& \quad \quad \text{Don't use any 4} \\
& \quad \quad \text{Solve two simpler problems:} \\
& \quad \quad \quad \text{count\_partitions}(2, 4) \\
& \quad \quad \quad \text{count\_partitions}(6, 3) \\
& \quad \text{Tree recursion often involves exploring different choices.}
\end{align*}
\]

Counting Partitions

The number of partitions of a positive integer \(n\), using parts up to size \(m\), is the number of ways in which \(n\) can be expressed as the sum of positive integer parts up to \(m\) in increasing order.

\[
\begin{align*}
count\_partitions(6, 4) & \quad \text{Recursive decomposition: finding simpler instances of the problem.} \\
& \quad \text{Explore two possibilities:} \\
& \quad \quad \text{Use at least one 4} \\
& \quad \quad \text{Don't use any 4} \\
& \quad \quad \text{Solve two simpler problems:} \\
& \quad \quad \quad \text{count\_partitions}(2, 4) \\
& \quad \quad \quad \text{count\_partitions}(6, 3) \\
& \quad \text{Tree recursion often involves exploring different choices.}
\end{align*}
\]